

מבחן SQL מתקדם

סיפור רקע – “מ Lager המכירות” (OPERATION SALES COVER) / מבצע Ciscoי מכירות)

תאריך האירוע הראשון: ינואר 2026
מקום ראשון: קראקס, ניו יורק (אזור התעשייה של פטארה, מבנה ישן שהוסב למרכז נתונים “פרטי”)
גורם מגלה: תחנת CIA בברוגטה, בשיתוף עם סוכנים מקומיים של ה-DEA שחשפו רשתות הלבנת הון של
קרטלי סמים בשיתוף עם גורמים איראניים.

בסוף 2025, בעקבות עלייה חדה בפעולותמודיעין איראנית-חיזבאללה בארה"ה הלטינית, ה-CIA הפעילה
מעקב רחב על שירותים ורטואליים ופיזיים באזורי קראקס. אחד השירותים – שנרשם תחת חברת “*Ventas*”
A.C. Globales. (חברת “מכירות גלובליות” – שם גנרי כמעט עד כדי חדשנות) – **הכיל מ Lager נתונים** שנראה
תמים להЛОוטין: מערכת ניהול מכירות ומשלוחים קלאסית.

המ Lager הועבר ליחידת הסיבר של המוסד, ואתם מצוות התקoor הטכני שנבחר לבצע את הניתוח הראשוני כחלק
משיתוף הפעולה המודיעיני בין ארה"ב לישראל.

המשימה שלכם:

לחקור את המ Lager באמצעות **שאילתות SQL ונקודות קצה (FastAPI)** שתתקבלו בשלד המערכת, ולהפיק
ממנו תוצאות שיאפשרו להבין:

- מי הגורמים המעורבים
- מה הדפוסים החרגים
- ואיפה “מסתתר” המידע הרגיש בתוך פעילות שנראית רגילה

המטרה שלכם היא אחת בלבד:

למשׁ שאלות SQL ולהציג תוצאות דרך נקודות קצה קיימות.

הקדמה – מבנה הפרויקט ואופן העבודה

במבחן זה אתם מקבלים **שלד מערכת Backend מוקן**, המבוסס על MySQL ומודול נתונים **FastAPI**.
המערכת נבנתה מראש כרשותם סביבת עבודה אמיטית בצוות פיתוח, שבו אתם נכנסים למערכת קיימת
ואתם נדרשם **להשלים לוגיקה חסירה – מבלי לשנות את התשתייה**.

מבנה הפרויקט (ברמה רעיונית)

הפרויקט מחולק לשכבות ברוורות:

שכבה תשתיית – INFRASTRUCTURE האחראית על:

- חיבור ל-MySQL
- ייצור בסיס הנתונים
- טעינת הטבלאות
- הרצת השרת

שכבה זו כבר כתובה ומוכנה, ואין לה מבחן.

שכבה DAL - DATA ACCESS LAYER

- מכילה פונקציות ריקות שעליים לממש.
- can כתבות שאילתות ה-SQL בלבד.

שכבה API - FAST API ROUTES

- מכילה נקודות קצה ריקות.
- can תחבורו בין ה-Endpoints לבין הפונקציה המתאימה מה-DAL.

חוקים חשובים (חוובה לקרוא)

חול איסור

- לשנות קבועים שקשורים לדocker
- לשנות קבועי חיבור למסד הנתונים
- לשנות קבוע היוזן של בסיס הנתונים
- להשתמש ב-ORM
- להוסיף ספריות חייזניות
- אין לכתוב SQL מוחז ל-dal.py

מה עליהם לעשות

- לכתוב SQL רק בתוך הפונקציות ב-`dal.py`.
- למש את הלוגיקה של כל `Endpoint` ב-`main.py`.
- כל `Endpoint` מחזיר JSON תקין.

תחול אוטומטי של המערכת

בעת הרצת השרת:

- החיבור למסד הנתונים נוצר אוטומטית
- בסיס הנתונים והטבלאות נוצרים אוטומטית
- אין צורך בהרצה ידנית של סкриיפטים
- אין צורך בהגדרות נספנות מצדכם

אם השרת עולה – סימן שהכול מוכן לעבודה.

AIR מתחילה לעבוד על הפרויקט – שלב אחרי שלב

כדי להתחיל את המבחן, תקבלו מאייתנו קוד מוקן מראש שנמצא ב-GitHub. הקוד כולל את כל התשתיה הדרישה (שרת, חיבור למסד נתונים, Docker וכו'), ואתם נדרשם רק להשלים את הלוגיקה החסורה בהתאם לשאלות המבחן.

חשוב מאד – אין לבצע GIT CLONE לריפוזיטורי זהה

במקום זאת, עליהם לבצע Fork – כולם יצור עותק אישי של הפרויקט תחת חשבון GitHub שלהם.

קישור לריפוזיטורי:

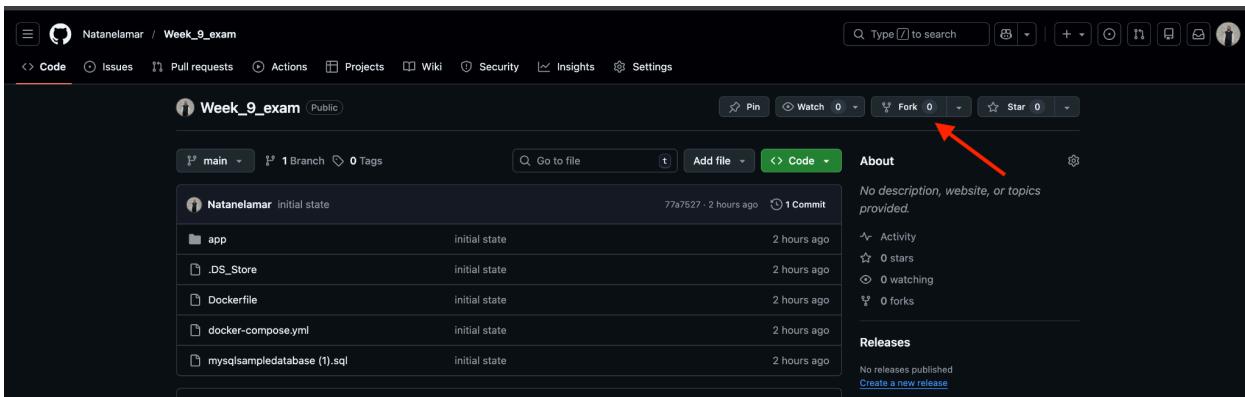
https://github.com/Natanelamar/Week_9_exam

שלב 1 – FORK לריפוזיטורי

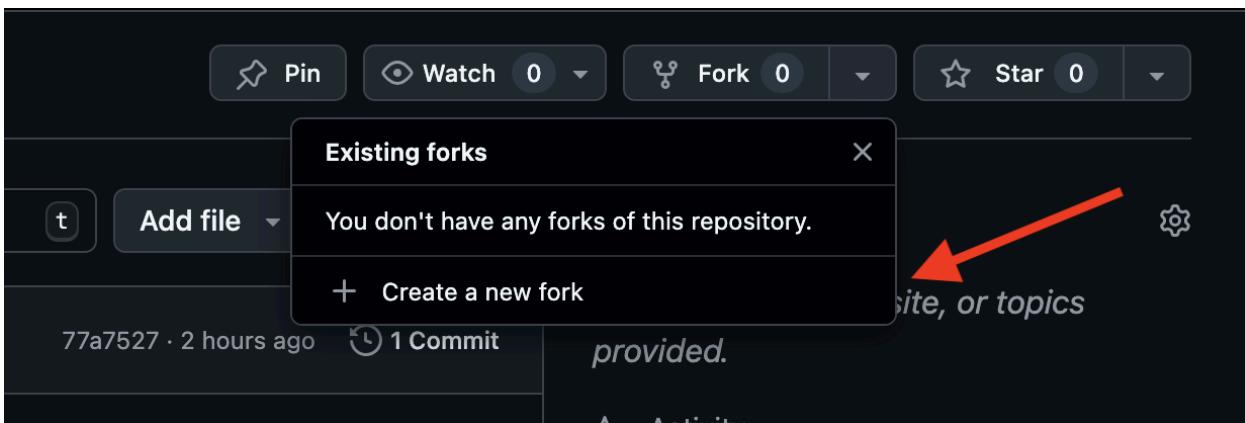
1. היכנסו לקישור של הפרויקט:

https://github.com/Natanelamar/Week_9_exam

2. לחצו על כפתור Fork (בצד ימין למעלה)



.3. תפוח לשונית ליצירת Fork חדש לחצן עלייה



.4. בחרו את המשתמש שלכם בתור הבעלים על העותק החדש

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner *	Repository name *
Choose an owner	Week_9_exam
Search owners	
<input checked="" type="checkbox"/> Natanelamar (repository already exists) <input checked="" type="checkbox"/> InnovatorsHub1	

Copy the main branch only
 Contribute back to Natanelamar/Week_9_exam by adding your own branch. [Learn more.](#)

Create fork



5. לאחר הפעלה, יוצר ריפורזיטורי חדש תחת החשבון שלכם בשם: week_15_DBSQL.

מעכשו – זהuproject שלכם.

שלב 2 – עבודה מקומית

לאחר שביצעתם Fork, ניתן לעבוד בצוות רגילה עם הריפורזיטורי שלכם.

1. היכנסו לריפורזיטורי שנוצר אצלכם
2. העתיקו את כתובת ה-HTTPS (Clone)
3. במחשב שלכם, פתחו טרמינל והריצו:

```
<git clone <YOUR_FORK_URL
```

(כאשר במקום <YOUR_FORK_URL> אתם רושמים את הקישור לריפו שלכם בפועל.)

4. היכנסו לתיקייתuproject:
`cd Week_9_exam`
5. מכאן ואילך אתם עובדים רק על הריפורזיטורי שלכם.

עבודה עם BRANCHES – חובה (10 נקודות)

במבחן זה נדרש לעבוד עם Git בצורה נקייה ומקצועית, בהתאם לסטנדרטים מקובלים בתעשייה.
לאחר ביצוע ה-Fork לריפודיטורי, עליו לגדיר מבנה Branches ברור ומוסדר שכוללים את הbranchים הבאים:

MAIN

- מכיל את הקוד הסופי לבדיקה
- רק זה יבדק

DEV

- עבודה Branch ראשי
- כל הפיתוח מתבצע דרכו
- בסוף העבודה, יש למזג (merge) אותו ל-`main`

לפיכך השאלות BRANCH

- מימוש שאלות SQL
- עבודה על `dal.py` ו-`main.py`
- מומלץ (אך לא חובה) ליצור Branch נוסף עבור כל שאלה - זהו הרג'ל עבודה נכון ומקצועי, המדמה עבודה בנסיבות אמיתיות.

YAML "יעודי" ל-K8S

- נפרד שמיועד אך ורק לעבודה על קבצי YAML ופריסת OpenShift Branch
- מטרתו להפריד בין:
 - פיתוח קוד אפליקציה
 - פיתוח תשתיות ופריסת

תהליך עבודה מומלץ

- עבודה שוטפת מתבצעת על Branches "יעוד"ים
- מיזוג (merge) ל-`dev`
- בסיום כל המבחן:
 - מיזוג `dev` אל `main`
 - `main` חייב להכיל את הפתרון המלא והסופי

בדיקות וניקוד

- יבדק שכל ה-Branches קיימים
- יבדק כי הקוד הסופי נמצא ב-`main`
- עבודה מסודרת עם Branches היא חלק מהציוון הכללי

שאליות SQL וניתוח נתוניים (8 נקודות – 8 נקודות פר שאילה)

בשלב זה עליו לתחקור את מאגר הנתונים שהועבר אליו, באמצעות **שאליות SQL** שתתמכשו במערכת הקיימת.

לכל שאלה:

- קיימת פונקציה ריקה בשכבה ה-DAL
- קיימת נקודת קצה (Endpoint) מתאימה
- עליהם למש את השאילה ולחבר אותה ל-`Endpoint` הרלוונטי

שאלה 1 – ליקוחות עם מסגרת אשראי חריגת

החזירו את שמות הליקוחות ומסגרת האשראי שלהם עבור ליקוחות אשר:

- מסגרת האשראי שלהם נמוכה מ-10,000
- או גבוהה מ-100,000

פלט נדרש:

- שם הליקוח
- מסגרת אשראי

שאלה 2 – הזמנות ללא העורות

החזירו את מספרי ההזמנות וההערות עבור הזמנות שאין להן הערות כלל (ערך **NULL**)

פלט נדרש:

- מספר הזמנה
- העורות
- מילן: לפי תאריכים של ההזמנה

שאלה 3 – חמשת הליקוחות האחרונים

החזירו את חמישת הליקוחות האחרונים לפי משרד כפי שהם מופיעים בטבלת הליקוחות.

פלט נדרש:

- שם הליקוח
- שם משפחה של איש הקשר
- שם פרטי של איש הקשר
- מילן לפי שם המשפחה

שאלה 4 – סיכון תשלוםם כולל, תשלום מינימלי ומקסימלי

החזירו:

- את סיכון התשלומים הכלול

- את ממוצע סכום התשלומים

עבור כל התשלומים במערכת.

פלט נדרש:

- סכום תשלום כולל
- ממוצע תשלום
- תשלום מינימלי
- תשלום מקסימלי

שאלה 5 – עובדים ומספרי טלפון של משרדים

הציגו עבור כל עובד:

- שם פרטי
- שם משפחה
- מספר הטלפון של המשרד שבו הוא עובד

פלט נדרש:

- שם פרטי
- שם משפחה
- טלפון משרד

שאלה 6 – ליקוחות ותאריכי משLOW

הציגו את שמות הליקוחות ואת תאריכי המשLOW של ההזמנות שלהם וממוצע של כמות ההזמנות הכללית .
יש לכלול גם ליקוחות שלא ביצעו הזמנות כלל.

פלט נדרש:

- שם הליקוח
- תאריך משLOW

- ממוצע כמות של הזמן הכללי

שאלה 7 – כמות פריטים בכל הזמן

החזירו את שמות הלקוחות ואת הכמות שהוזמנה בכל הזמן.

פלט נדרש:

- שם הלקוח
- כמות שהוזמנה
- מין לפי שם הלקו

שאלה 8 – ליקוחות חדשניים לפי שם משפחה

החזירו:

- שם הלקוח
- שם איש המכירות
- סכום התשלומים הכלול

עבור ליקוחות שם הפרטி של איש הרקע שלהם מכיל את המחרוזת 'Mu' או 'ya'.

פלט נדרש:

- שם הלקוח
- שם איש המכירות
- סכום תשלוםם כולל
- יש למין את התוצאות לפי סכום התשלומים הכלול **בסדר יורד**.

דגשים כלליים

- יש להציג נתונים מדויקים בהתאם לפט הנדרש
- אין חשיבות לשמות העמודות, אלא לערכים
- שימוש נכון ב-NJOIN הוא חלק מהפתרון

- סדר התוצאות חשוב רק כאשר צין במדויק

פרישה לאופן שיפט (8 נקודות)

בשלב זה עליהם לפרש את האפליקציה שבניתם לסביבת SHIFTOPEN, באמצעות קבצי YAML שתכתבו בעצמכם.

קובצי YAML – מה נדרש מכם

עליכם לכתוב את כל קבצי ה-YAML הדרושים על מנת:

- להריץ את אפליקציית ה-[FastAPI](#)
- להריץ מסד נתונים MySQL
- לאפשר תקשורת בין האפליקציה למסד הנתונים
- לחשוף את האפליקציה החוצה כרשותנית יהיה לגשת אליה מהדף

מספר קבצי YAML כולל:

צפיה של לפחות 4 קבצים (בהתאם לפתרון שלכם)

אין פירוט אילו משבבים לבדוק נדרש.
עליכם להבין בלבד אילו רכיבים יש להגדיר כדי שהמערכת תעבור מקרה לקרה.

צלומי מסך (2 נקודות)

בסיום הפרישה, עליכם להעלות 2 צילומי מסך:

1. כל ה-Pods רצים

a. צילום מסך המראה שכל ה-Pods בפרויקט במצב **Running**.

.ב. יש לוודא שניתן לראות את שמות ה-Pods והսטטוס שלהם.

2. האפליקציה רצה בדף

צילום מסך של האפליקציה פתוחה בדף.

- a. יש להציג את **שורת ה-URL המלאה** OpenShift בבירור את הכתובת שנוצרה על-ידי (localhost)
- b. יש לראות בבירור את הכתובת שנוצרה על-ידי (localhost) (לא
- c. יש לגשת ל-Route של האפליקציה (לא